MASS: Middleware-based Adaptive Software System

Qianxiang Wang
School of Electronics Engineering and Computer Science, Peking University
Key laboratory of High Confidence Software Technologies (Peking University), Ministry of Education
Beijing, China, 100871
wqx@pku.edu.cn

Abstract

Software that runs in the network environment is facing with many challenges. One of them is the dynamic feature of the network. To keep the quality of service, the service entity needs to adapt promptly to the changing environment. The traditional system software, operating system, which is used mainly to manage the hardware resources of computer, doesn't provide enough support for application adaptation at runtime. This paper introduces a middleware based approach that enhances the adaptability of software system. The key contribution of the proposed approach is that it clearly separates the adaptation program from the function program at runtime. The past, current and future work of MASS are introduced also.

1. Motivation

In 1945, John Von Neumann proposed the technique of “stored program”. Program, which is composed of instructions, is used to control the simple hardware. When the computer function needs to be changed, the owner needs not to hand-wire the actual hardware, but just change the program. The hardware keeps stable, although the computer function has changed.

Once the program is loaded into the main memory, it keeps stable also, before it is removed from the main memory. Program is changed outside the memory, offline.

When software is introduced into more systems, the phenomenon of “Software as Service” emerged. Compared with the traditional “computing oriented software”, one of the great challenges to the “servicing oriented software” comes from servicing environment. The service environment is dynamic, in other words, many service related factors keep changing during the service, e.g., client number, client role, servicing resources, network bandwidth. These changing factors are potential risks to quality of service. For example, limited hardware resources often lead to a long response time, too many client requests may lead to low performance, and intrusion request may even lead to the deny of service.

This situation requires the service with the following abilities:

(1) When the service environment changes, to keep the quality of service, the service entity needs to be adjusted.

(2) The adjusting process needs to occur at the servicing time. Considering the adjusting process need to be very smart to the changing environment, automatically adjusting is expected.

The key problem then arises: because the program keeps stable at the servicing time, which part of service entity should be changed?

There are some existing research areas that have touched the similar issue, e.g., Fault tolerant, Intrusion detection, Load sharing, and adaptive network routing. All of those areas have got well solution to some specific changing environment.

Meanwhile, in recent years, some related concepts are proposed, e.g., autonomic computing [1], self adaptive software [2], and software cybernetics [3].

In recent year, more and more service systems are developed based on middleware, which is used widely to enhance the software quality in the internet environment. In this paper, we introduce MASS: a Middleware based Adaptive Software System. The key contribution of MASS is that it clearly separates the adaptation program from the function program at runtime.

The rest of the paper is organized as follows: section 2 introduces the MASS, while section 3 describes the roadmap of MASS, including the past, current and future work of MASS.
2. MASS Architecture

In this paper, we mainly introduce a project in Peking University: MASS. The goal of this project is to enhance the adaptability of service systems, so as to provide good service in the continually changing environment, with the support of new developed middleware technology.

2.1 Architecture

Network based service system is usually composed of hardware and software. Nowadays, most of software is divided into multiple layers. A typical architecture of service system is shown in Figure 1(a), in which software is composed of three layers: operating system, middleware and application. MASS focuses on middleware and application.

To enhance the adaptability of service, we divide the application further, and gain a novel structure of application. In the new structure, application is divided in two layers: function program layer and adaptation program layer. See Figure 1(b).

Function program is application program that can affect directly the functional relationship between client’s input and output. It is the “hardware” of application, and responsible for the basic function of service.

Adaptation specification is part of application that monitor and control the configuration of function program. It is the “software” of application, and responsible for the quality of service.

2.2 Inside the Adaptation Specification

Adaptation specification is used to manage the function program. It monitors the behavior and environment of function program at servicing time, do some deep analyzing, and then based on the analyzing, makes adaptation decision, and adjusting the function program, or related elements.

- Monitoring
  “You can't control what you don't measure” [4]. Monitoring technology is widely used in many kinds of software. Starting from early 1960s with the advent of debuggers, it was used for debugging and testing, correctness checking, security, dependability, performance enhancement, performance evaluation, and controlling [5]. Recent taxonomy work shows runtime software monitoring has been used for profiling, performance analysis, software optimization as well as software fault-detection, diagnosis, and recovery [6]. The key issues including: “What should be monitored”? “How to monitor the system”? “Who responsible for the monitor”? “What is monitor friendly application”?

- Analyzing
  Analyzer is used to process different kinds of monitored event. The basic analyzing approach is to validate the monitored events against the pre-specified constraint. The analyzer is responsible for: 1) Constraint violation detection, e.g., wrong parameter value region, wrong message order, over weighted load, and response time that exceeds some limit are all typical constraint violations; 2) Risk prediction. This function is used to
predict potential risks, such as load overweight, memory leak, and deadlock, so as to adjust the system in time; 3) Problem determination. This function is used to reckon the reason of service behavior deviation. It is very important for deciding what to do further, so as to guarantee the service quality.

Some high level analyzing work include: malicious pattern matching, attack prediction, and adaptation condition determining.

- Adjusting
  This is the most important part of adaptation program. It relies strongly on the interfaces that are provided by the low level systems, such as the function program and the middleware.

  There are two basic kinds of adjusting APIs: API for adjusting function program and API for adjusting environment.

  For function program adjusting, the target objects include: algorithm, object, component, service, and relationship between them. From another view, some typical adjusting ways include: target update, target migration, and target reconfiguration.

  For environment adjusting, the target objects include: client connection numbers, instance number, and resource (CPU, memory, etc.).

  The adjusting is goal oriented, and the adjusting result usually needs to be evaluated. When there are multiple goals, coordination between multiple goals is necessary.

  More adjusting related issues include: how to cope with chained adaptation? How to keep adapt process safe? How to avoid inconsistency state? There exist some good research results toward those issues [7] [8].

- Management Information Base
  Management Information Base is a special part in the adaptation specification. Actually, it is not a real program, but base of constraints, rules, policies and logs etc. In adaptation process, it has strong affect to the analyzing work and evaluation work. Adaptation rules are keys for different kinds of adaptation. It is usually pre-specified by administrator, and can be implemented inside program, or stored inside the base. More effective rules should be mined from the real servicing process and vast log of the servicing status.

2.3 Why Middleware based?

  The traditional system software, operating system, which is used mainly to manage the hardware resources of computer, doesn’t provide support for application adaptation at runtime. On the contrary, as a new kind of system software, middleware locates between operating system and application. It is used to make applications work together and provide services with better quality.

  Middleware is usually based on client/server paradigm. At the beginning, it is used to make network based programming process easier, e.g. RPC, MOM. It is used later to improve the quality of service, e.g., TP monitor, security monitor. Anyway, middleware has is a good platform for orchestrating the function code.

  Current middleware has some adaptation ability and provide a good foundation. Especially, as the most important middleware, application server provides specific interface for management of component at runtime, which provides more adjusting opportunities.

3. Roadmap

  MASS is based on our former work of PKUAS (PeKing University Application Server). The adaptation feature was soon introduced into PKUAS. This section introduces the past, current and future work of MASS.

3.1 Past Work

  PKUAS started in 2001. It is an application server that compliant with J2EE specification, has componentized, plug-in architecture. It implements all the standard J2EE component containers and provides multiple interoperable protocols and abundant enterprise common services [9]. PKUAS has been successfully applied in the construction of enterprise information platform and the education and the traffic domains.

  As the first work related with adjusting, online software component updating was explored during the development of PKUAS. In [14], we presented a component-based approach to online software evolution, as well as the implementation of our approach, that is based on the Java programming language and J2EE compliant application servers. Initial rule model for adaptation was touched also [15].

  From 2004, software cybernetics, a series of adaptation related issues were explored [10-13]. Some of the research results were introduced. While many other research groups that focus on issues in testing [3], MASS focuses on issues at runtime.

3.2 Current Work

  Current research work of MASS is focused on monitoring. A monitoring model and monitoring
framework are proposed in [16]. The monitoring model involves five parts of service system: responses message, application, resource, requests message, and management operations. While monitoring model aims to answer the question of “what should be monitored”, the monitoring framework tries to answer the question of “how to monitor”. The framework is composed of four parts: distributed probes, agent, central analyzer and high-level representation. Constraint specification, which can be used to monitor request and reply messages is deeply explored also.

Main adjusting work is moved from online updating to online migration. Different with the traditional migration of process, we focus on online migration of service, especially for high performance service system.

3.3 Future Work

While the research on adjusting and monitoring will be kept continually, the research on analyzing will be enhanced soon. Analyzing is the bridge from monitoring to adjusting. A well explored analyzing research work will lead to an automated adaptation process.

The key work of analyzing are various kinds of mining: 1) Mining new constraint, e.g., conditions of adaptation; 2) Mining new pattern, e.g., usage preference, time distribute, area distribute, intrusion; 3) Mining new rule: causal relationship between input (environment, message, use, admin) and software behavior.

Acknowledgement

This paper is supported by the National High-Tech Research and Development Plan of China No. 2006AA01Z175, the National Grand Fundamental Research Program of China No. 2005CB321805, and Key National Science Foundation of China, No. 90412011.

References